AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. (original) A method of executing a single instruction parallel multiply-add function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel multiply-add instruction;

providing the processor with a first, a second and a third value, wherein each of the values comprises two or more operand components;

multiplying first operand components of the first and the second values to generate a first intermediate value;

multiplying second operand components of the first and the second values to generate a second intermediate value;

adding a first operand component of the third value to the first intermediate value to generate a first result value;

adding a second operand component of the third value to the second intermediate value to generate a second result value;

storing the first result value in a first portion of a result location; and

storing the second result value in a second portion of the result location.

2. (original) The method of claim 1, wherein the first, second and third values are stored in respective source registers of the processor specified by the parallel multiply-add instruction, and the first and the second result values are stored in a destination register of the processor specified by the parallel multiply-add instruction.

3. (original) The method of claim 2, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

4. (original) The method of claim 1, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction every 2 cycles.

5. (withdrawn) A method of executing a single instruction conditional pick function on a processor, the method comprising:

providing the processor with an opcode indicating a conditional pick instruction;

providing the processor with a first, a second and a third value;

comparing the first value to a reference value;

determining, based upon the comparing, whether the first value is equal to the reference value;

storing the second value in a result location if the first value is equal to the reference value; and

storing the third value in a result location if the first value is not equal to the reference value.

6. (withdrawn) The method of claim 5, wherein the first, second and third values are stored in respective source registers of the processor specified by the conditional pick instruction, and the second and the third values are stored in a destination register of the processor specified by the conditional pick instruction.

7. (withdrawn) The method of claim 5, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

8. (withdrawn) A method of executing a single instruction parallel averaging function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel averaging instruction;

providing the processor with a first and a second value, wherein each of the values comprises two or more operand components;

adding first operand components of the first and the second values to generate a first intermediate value;

adding second operand components of the first and the second values to generate a second intermediate value;

incrementing the first intermediate value by one to generate a third intermediate value;

incrementing the second intermediate value by one to generate a fourth intermediate value;

shifting the third intermediate value to generate a first result value;
shifting the fourth intermediate value to generate a second result value;
storing the first result value in a first portion of a result location; and
storing the second result value in a second portion of the result location.

9. (withdrawn) The method of claim 8, wherein the first and the second values are stored in respective source registers of the processor specified by the parallel averaging instruction.

10. (withdrawn) The method of claim 8, wherein the first and the second result values are stored in a destination register of the processor specified by the parallel averaging instruction.

11. (withdrawn) The method of claim 10, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

12. (withdrawn) The method of claim 8, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

13. (withdrawn) A method of executing a single instruction parallel shift function on a processor, the method comprising:
providing the processor with an opcode indicating a parallel shift instruction;
providing the processor with a first and a second value, wherein each of the values comprises two or more operand components;
shifting the first operand component of the first value by a number of bits equal to a value of the first operand component of the second value to generate a first result value;
shifting the second operand component of the first value by a number of bits equal to a value of the second operand component of the second value to generate a second result value;
storing the first result value in a first portion of a result location; and
storing the second result value in a second portion of the result location.

14. (withdrawn) The method of claim 13, wherein the first and the second values are stored in respective source registers of the processor specified by the parallel shift instruction.

15. (withdrawn) The method of claim 13, wherein the first and the second result values are stored in a destination register of the processor specified by the parallel shift instruction.

16. (withdrawn) The method of claim 15, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

17. (withdrawn) The method of claim 13, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

18. (Previously Presented) A processor comprising:

a first and second multiplier paths;

a first and second adder paths; and

wherein the processor supports a parallel multiply-add instruction, the parallel multiply add instruction executable to cause the processor to,

in parallel, route a first component of a first operand and a first component of a second operand to the first multiplier path and a second component of the first operand and a second component of the second operand to the second multiplier path,

in parallel, route output of the first multiplier path and a first component of a third operand to the first adder path, and output of the second multiplier path and a second component of the third operand to the second adder path, and

store output of the first adder path at a first location and output of the second adder path at a second location.

19. (Previously Presented) The processor of claim 18, wherein the parallel multiply-add instruction operates on either integer or fixed point operands.

20. (Previously Presented) The processor of claim 19, wherein the results of the parallel multiply-add instruction are saturated.

21. (Previously Presented) The processor of claim 19, wherein the processor provides multiple saturation modes.

22. (Previously Presented) The processor of claim 18,

wherein the processor further supports a conditional pick instruction, the conditional pick instruction executable to cause the processor to compare a first value to zero and to copy either a second value or a third value to a destination location depending on the comparison.

23. (Previously Presented) The processor of claim 18,

wherein the processor further supports a parallel averaging instruction, the parallel averaging instruction executable to cause the processor to average a first operand's first component and a second operand's first component, and, in parallel, to average the first operand's second component and the second operand's second component.

24. (Previously Presented) The processor of claim 18,

wherein the processor further supports a parallel shift instruction, the parallel shift instruction executable to cause the processor to logically shift a first portion of a first value in accordance with a first portion of a second value, and, in parallel, shift a second portion of the first value in accordance with a second portion of the second value.

25. (Currently Amended) The processor of claim 18 wherein the processor further supports a parallel power instruction, the parallel power instruction executable to cause the processor to,

raise a first component of a first operand to a power indicated in a first component of a second operand and, in parallel, raise a second component of a the first operand to a power indicated in a second component of the second operand.

26. (Previously Presented) The processor of claim 18 wherein the processor further supports a parallel reciprocal square root instruction, the parallel reciprocal square root instruction executable to cause the processor to,

determine a reciprocal square root of an operand's first component and, in parallel, determine a reciprocal square root of the operand's second component.

27. (Previously Presented) A computer program product encoded on one or more machine-readable media, the computer program product comprising:

an instruction sequence, the instruction sequence including an instance of a parallel multiply add instruction;

the instance of the parallel multiply add instruction having an at least four operand instruction format,

wherein execution of the parallel multiply add instruction

causes generation of a first product from a first operand's first component and a second operand's first component, in parallel with generation of a second product from the first operand's second component and the second operand's second component,

causes generation of a first sum from the first product and a third operand's first component, in parallel with generation of a second sum from the second product and the third operand's second component, and

causes the first sum to be stored in accordance with a fourth operand's first component and the second sum to be stored in accordance with the fourth operand's second component.

28. (Previously Presented) The computer program product of claim 27, wherein the operands include one or more of a fixed point format and an integer format.

29. (Previously Presented) The computer program product of claim 27, wherein the first components correspond to the high order bits of the respective operands and the second components correspond to the low order bits of the respective operands.

30. (Previously Presented) An apparatus comprising:

a plurality of registers; and

means for performing, in response to a single instruction instance, a parallel multiply add operation, the parallel multiply add operation causing generation of a first product and a second product in parallel, and causing generation of a first sum and second sum in parallel, wherein an input value for the first sum includes the first product and an input value for the second sum includes the second product.

31. (Previously Presented) The apparatus of claim 30, further comprising a plurality of multipliers and adders.

32. (Previously Presented) The apparatus of claim 30, wherein the parallel multiply add operation further causes storing of the first sum in a first portion of a first of the plurality of registers and storing of the second sum in a second portion of the first register.

33. (Previously Presented) A method of executing an instruction instance comprising:

generating a first product and a second product in parallel, wherein the first product is from a first and second value and the second product is from a third and fourth value; and

generating a first sum and a second sum in parallel, wherein the first sum is from the first product and a fifth value and the second sum is from the second product and a sixth value.

34. (Previously Presented) The method of claim 33, wherein the first and third values respectively are first and second portions of a first operand, the second and fourth values respectively are first and second portions of a second operand, and the fifth and sixth values respectively are first and second portions of a third operand.

35. (Previously Presented) The method of claim 33 further comprising storing, in parallel, the first sum in a first location and the second sum in a second location.

36. (Previously Presented) The method of claim 35, wherein the first location is a first portion of a destination register and the second location is a second portion of the destination register.

37. (Previously Presented) The method of claim 33 wherein the instruction instance is executed by a pipelined processor that performs operations for the instruction instance in 2 cycles.

38. (Previously Presented) The method of claim 33 embodied as a computer program product encoded in one or more machine-readable media.

39. (Previously Presented) The processor of claim 18, wherein the first store location is a first part of a register and the second store location is a second part of the register.

40. (Previously Presented) The processor of claim 18, wherein the first store location is a first register and the second store location is a second register.

41. (Previously Presented) The processor of claim 18, wherein the first and second multiplier paths are embodied as distinct functional units.

42. (Previously Presented) The processor of claim 18, wherein the first and second adder paths are embodied as distinct functional units.

43. (Previously Presented) The processor of claim 23 further comprising:

a plurality of adder paths; and

a plurality of shifter paths;

wherein the parallel averaging instruction, when executed, causes the processor to,

route the first operand's first component and the second operand's second component to a first of the plurality of adder paths, and, in parallel, route the first operand's second component and the second operand's second component to a second of the plurality of adder paths;

after propagation delay, route output of the first adder path and a one value to a third of the plurality of adder paths, and, in parallel, route output of the second adder path and a one value to fourth of the plurality of adder paths;

after propagation delay, route output of the third adder path and a first control value a first of the plurality of shifter paths, and, in parallel, route output of the fourth adder path and a second control value to a second of the plurality of shifter paths.